12_18 作業系統小考筆記

Converted via https://cloudHQ.net
Tags: 小考筆記
Created: December 16 2017 +00:00, 02:37:50   Modified: December 17 2017 +00:00, 15:11:24
Note URL: https://www.evernote.com/Home.action#n=950663f3-92bd-4312-ab05-511dcd506d23&b=346647db-96e5-4f2f-b8e5-82828ea46ccd&ses=4&sh=1&sds=5

[CH6]
解決**critical-section**問題的方法:

solution 需滿足的三個條件:

- mutual exclusion:
- 任一時間點，只允許一個process進入critical section內活動
- progress: 同時滿足2個要件
- 不想進入 critical section 的process不可以阻礙其他process進入critical section，即不可參與進入critical section 的決策過程
- 必須在有限的時間從想進入critical section的process中，挑選其中一個process進入critical section,**隱含No Deadlock**
- bounded waiting 自process提出 進入critical section的申請到獲准進入critical section的等待時間是有限的。即若有n個processes想進入，則任一process 至多等待n-1次即可進入，**隱含No starvation**

Peterson's Solution-軟體方式解決

```
do{
    flag[i] = TRUE;
    turn = j ; // 禮讓的概念

    while(flag[j]&&turn==j); // 想進去且輪到他進去
    CRITICAL SECTION
    flag[i] = FALSE;
    REMAINDER SECTION
} while(TRUE);
```

The structure of process Pi in Peterson's solution

假設有兩個process,共享兩個變數

[data structure]
int turn (代表輪到誰進入critical section)
ex:
turn==i process Pi is allowed to execute in its critical section

boolean flag[2] (代表誰準備好進入critical section)
ex:
flag[i] is true Pi is ready to enter its critical section

1 Mutual exclusion
若Pi與Pj皆想進入自己的Critical Section，代表 flag[i] == flag[j] == true，且分別執行到turn=i及turn=j 之設定，先後順序不同，turn的 僅會是i或j, 不會兩者皆是
2 Progress
若Pi 不想進Critical Section ,則表示flag[i] = false。此時若Pj想進入自己的Critical Section,必可通過while(flag[i]andturn==i)do no-op這個空迴圈而進入CS，不會被Pi阻礙。
3 Bound-waiting
Pi 離開CS 後又企圖立刻進入自己的CS,此時Pi一定會執行turn=j,使得Pi無法再搶先於Pj進入自己的CS。所以Pj至多等待一次即可進入CS。

[hardware solution]
使用硬體 就不會有"synchronization"的問題
=>因為不會被中斷

Atomic (不被中斷) : TestAndSet()

```
boolean TestAndSet(bool &lock)
{
    bool value = lock;
    lock = TRUE;
    return value;
}
```

解說: excute atomically
return the value of "lock"
and set "lock" to TRUE

一開始 初始lock為false(0) , 第一個執行TestAndSet() 的process 會傳回 false,因此進入**critical section** ,在呼叫TestAndSet的同時 會將lock設成1 ,使得其他process無法進入， 當做完critical section,lock設成0,讓其他process也有機會進入critical section

"3條件" 符合狀況
mutual exclusion ? Yes
Progress ? Yes
Bounded-Wait ? No!
Why not?
因為是用搶的 看誰先call TestAndSet

mutex(mutual exclusion) locks
acquire(): acquires the lock
release(): releases the lock

```
acquire(){
    while(!available)
    ; /busy wait/
     available = false;
}
```

```
release{
    available = true;
}
```

缺點: requires busy waiting -> wastes CPU cycles
real multiprogramming system,where a single CPU is shared among many processes
優點: no context switch(耗時) is required when a process must wait on a lock
good for multiprocessor system

Semaphore
A tool to generalize the synchronization problem.
easy to solve , but no guarantee for correctness

a record of how many units of a particular resources are available

if #record = 1 -> binary semaphore,mutex lock
if #record > 1 -> counting semaphore

classical problems of synchronization
purpose:用來驗證解決synchronization的解法有沒正確

- Bounded-Buffer (Producer-Consumer) Problem
- Reader-Writers Problem (檔案，資料的操作)
- Dining-Philosopher Problem

Bounded-Buffer Problem
buffer:
空的時候 => consumer等
滿的時候 => producer等
[CH5]

processor affinity:
讓processor盡可能在同一個processor上處理
避免cache invalidatingc和repopulating

soft affinity:
可讓process在processor之間轉移
hard affinity:
可讓process "不能"在processor之間轉移
在windows下　可用affinity指令控制process要在 個processor上執行

Load　Balance
兩種方法：
- Push migration　（高至低）
load 高的processor主動將工作 push　給　load 低的人
- Pull migratiion
load 低的　主動向　load 高的　pull 工作來做

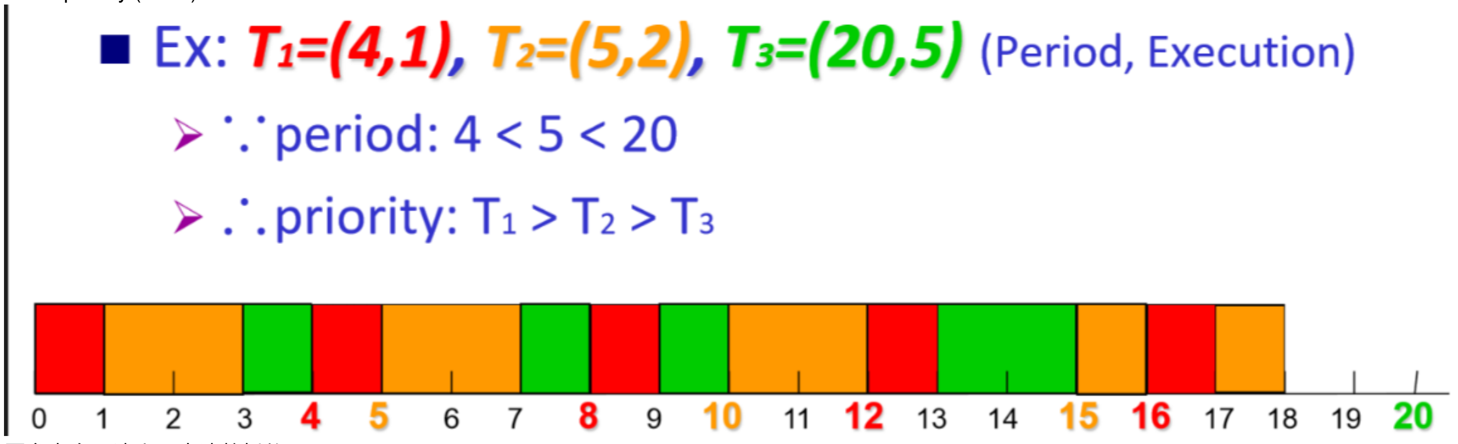
load 高的時候：pull
load 低的時候：push

Real time: 在deadline之前完成工作，does not mean speed

Real-time Scheduling:
Soft real-time requirements:
盡量避免　missing the dealine
ex: Multimedia streaming
Hard real-time requirements:
保證不會miss dealine
ex: 核電控制，車子

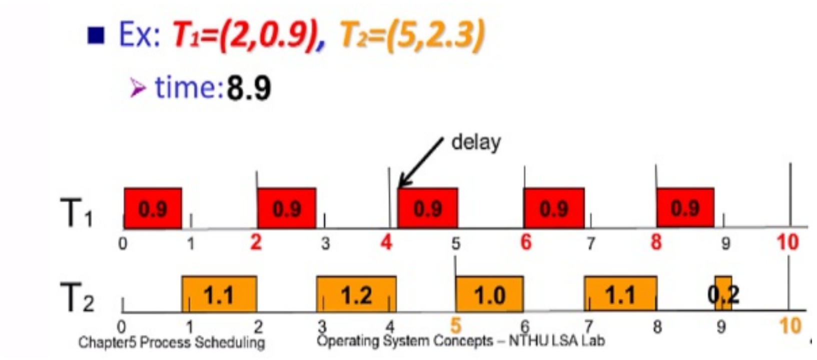
［ａｌｇｏｒｉｔｈｍ］
ready : 什麼時候進到系統要執行/execution(cpu burst)/period(deadline)
假設規律

Rate-Monotonic(單一性)(RM)　alg : // 很常被用到
–shorter period -> higher priority
-fixed-priority (static)


- Ex: $T_1=(4,1)$, $T_2=(5,2)$, $T_3=(20,5)$ (Period, Execution)
  - ∵period: 4 < 5 < 20
  - ∴priority: $T_1 > T_2 > T_3$

圖參考自：清大周志遠教授的ppt

Earliest-deadline-first(EDF) alg: // 更常
ex:寫作業，交作業
-earlier dealine -> higher priority
-Dynamic priority


Ex: T1=(2,0.9), T2=(5,2.3)
time:8.9
delay
T1
0.9
T2
1.1
1.2
1.0
1.1
0.2
0 1 2 3 4 5 6 7 8 9 10
Chapter5 Process Scheduling
Operating System Concepts – NTHU LSA Lab

圖參考自：清大周志遠教授的ppt

[ppt 重點：] -- Mutiple Processor Scheduling

Asymmetric multiprogramming
a single processor has all **scheduling decisions I/O processing,and other system activities**
[master server]
the other processor excutes **only user code**
simple:only one processor accesses the system data structures ~~data sharing~~

Symmetric multiprogramming (SMP)
-each processor 自行做schduling
-all processors may be in a common ready queue,or each processor may have its own private queue of ready processes
-schduling 在執行時　使用　schduler examine ready queue and select a process to excute
complex: schduler must be programmed carefully when multiple processor trying to access and update a common data structure

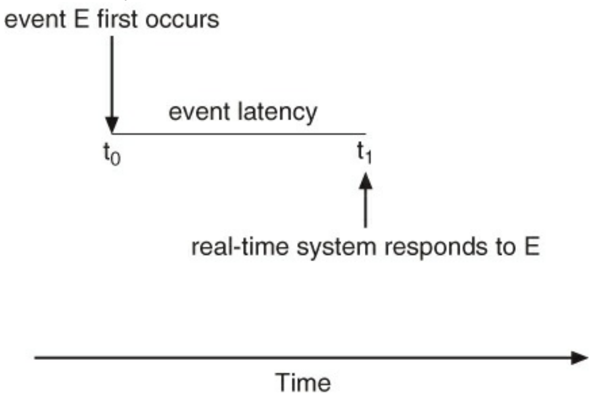
muticore processors

threads 在同一個 chip 裡
跟傳統的multi-processor比，更快更省power

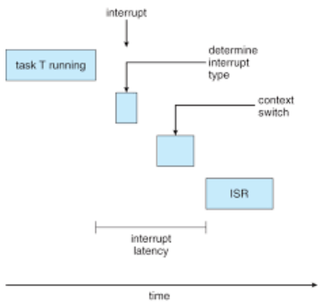
Real time CPU Scheduling

- soft real-time system: 系統盡量幫忙達成目標，但不保證
- hard real-time system: 所有的工作，都必須在deadline 前完成

minimizing latency:
Event latency :



兩種types latency 影響real-time system 的效能
interrupt latency:



dispatch latency:

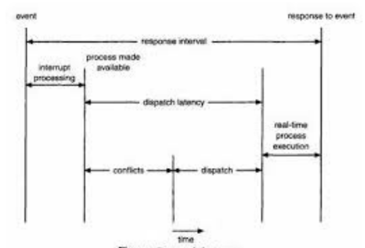


Figure-Dispatch latency

Priority Based Scheduling:

with preemption

periodic:

0<=t<=d<=p

- t: fixed processing time
- d: deadline
- p:period

rate: 1/p

Proportional Share Scheduling

proportinal share schedulers operate by allocating T shares among all application

an application can receive N shares of time

the application will have N/T of the total processor time

ex:係

a total of T = 100 shares　 分給　A B C

A is assigned 50 shares

B is assigned 15 shares

C is assigned 20shares

in conjuction with [admission control]

[admission control]　有足▯資源就給　沒有就deny

Algorithm Evaluation

CPU utilization: CPU 使用時間

Troughput: 單位時間完成的工作量

Turnaround time: 進去process到出來花的時間

Response time : 自使用者命令交付給系統到第一個回應所需的時間　// for time sharing system,user interactive app

deterministic modeling

takes a particular predetermined workload 再用各個如下的alg去評估

ex:FCFS SJF RR(quantum=?)

Queueing models

little's formula: n = 入　＊　W (exponential)

n: average queue length

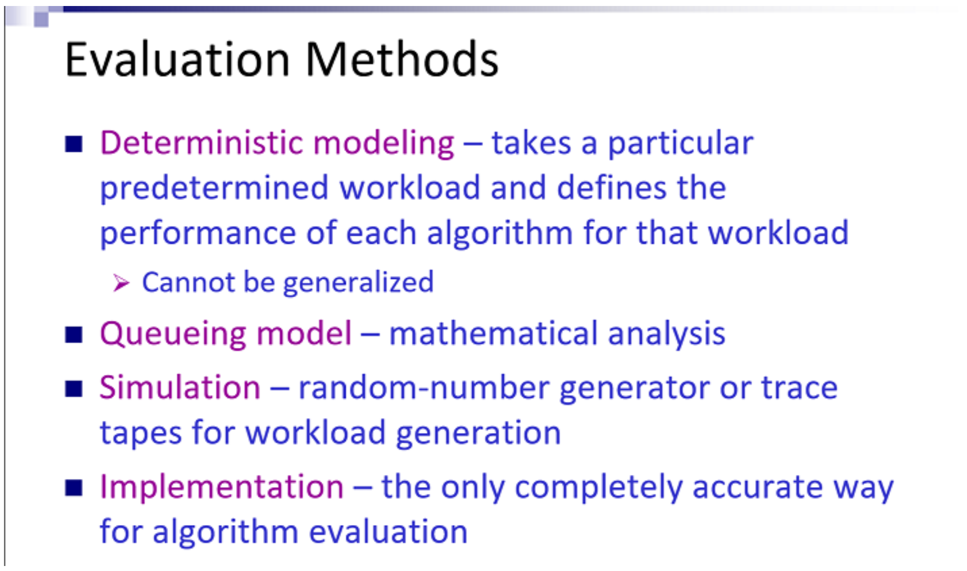
W: average waiting time in the queue

入: aveage arrival rate for new process in the queue(每秒幾個processes)

ex: 數學公式去分析

simulations
random number generator or trace tapes for workload generation

implemention
最準確的方式



Evaluation Methods

- Deterministic modeling – takes a particular predetermined workload and defines the performance of each algorithm for that workload
  - Cannot be generalized
- Queueing model – mathematical analysis
- Simulation – random-number generator or trace tapes for workload generation
- Implementation – the only completely accurate way for algorithm evaluation

參考自　清大開放式課程的ppt